LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-PROC-680910

The Improved Simple Corner Balance Method and Efforts to Enhance its Computational Performance

P. G. Maginot, P. N. Brown, A. J. Kunen, T. S. Bailey

January 19, 2016

2016 ANS Annual Meeting
New Orleans, LA, United States
June 12, 2016 through June 16, 2016

Disclaimer

The Improved Simple Corner Balance Method and Efforts to Enhance its Computational Performance

Peter G. Maginot, Peter N. Brown, Adam J. Kunen, Teresa S. Bailey

Lawrence Livermore National Laboratory, Livermore, CA 94551
maginot1@llnl.gov, brown42@llnl.gov, kunen1@llnl.gov, bailey42@llnl.gov

INTRODUCTION

The diamond difference (DD) spatial discretization has long been a work horse of the neutron transport community. For nearly as long as DD has been used in neutron transport, it has been observed that DD produces undamped, non-physical angular fluxes in spatially under resolved problems [1] and can fail in the thick diffusion limit [2]. A long list of alternative spatial discretizations, all with better physics properties: accuracy, robustness, and/or preservation of the thick diffusion limit, have been proposed as alternatives to DD.

Unfortunately, these alternative spatial discretizations also require significantly more floating point operations (FLOPs) per unknown compared to DD. Historically, neutron transport problems of interest have been able to be resolved sufficiently that DD is accurate enough and FLOPs have been the computational performance bottleneck controlling time to solution. Thus, in practice DD has been and remains a preferred spatial discretization for neutron transport simulations.

Trends in computing architecture are leading to a new paradigm where data movement, not FLOPs will control time to solution [3]. This presents an opportunity to improve neutron transport physics by reconsidering more FLOP intensive spatial discretization than DD without incurring a time to solution penalty. In this work, we consider the improved simple corner balance (ISCB) spatial discretization, a unique alternative to DD that has the same memory footprint as DD, and detail efforts to improve its computational performance.

IMPROVED SIMPLE CORNER BALANCE METHOD

The ISCB scheme is a member of the family of corner balance spatial discretizations [4]. Like all corner balance schemes, ISCB is defined by maintaining particle balance on individual corners that are the constituents of a larger spatial zone. However, unlike the simple corner balance or upstream corner balance methods [4] ISCB does not individual corners of the larger spatial zone are composed of the same material or have the same dimensions; ISCB treats each corner as having a unique macroscopic cross section and dimensions [5]. As implemented in ARDRA [6], there is one unknown corner average flux per material zone. This makes the ISCB of ARDRA unique as a corner balance scheme; the ISCB of ARDRA has an identical memory footprint as DD when applied to the same spatial mesh. In contrast, other corner balance schemes [4] require multiple unknowns per material zone, resulting in a much larger memory footprint than DD when applied to the same spatial mesh.

The ISCB scheme on 3-D Cartesian bricks is defined by integrating the mono-energetic, steady state, fixed source transport equation over the 8 individual material zones, corners in the notation of [4, 5], that define a single "super" zone for a given discrete-ordinate direction and energy group. The 8 corner balance equations are exact, but have more unknowns than equations. To close the system of equations, an upwinding condition is used on the super zone boundaries,

$$\psi_{f,c} = \begin{cases} \psi_c & \vec{\Omega} \cdot \vec{n}_f > 0 \\ \psi_b & \vec{\Omega} \cdot \vec{n}_f < 0 \end{cases} , \tag{1}$$

where for corner c, the corner average unknown angular flux is ψ_c, $\vec{n}_f$ is the outward directed normal of face f (relative to corner c), $\psi_{f,c}$ is the average angular flux on face f, and ψ_b is a boundary condition or previously determined outflow angular flux from an adjacent super zone solution. On faces interior to a super zone, $\psi_{f,c}$ is defined as

$$\psi_{f,c} = \frac{\psi_c + \psi_{c'}}{2} , \tag{2}$$

where $\psi_{c'}$ is the corner average flux of the corner adjacent to c across face f. The definition of Eq. (2) requires the solution of 8 linear equations to yield the 8 unknown corner average fluxes of a given ISCB super zone. The resulting matrix is not dense, but is also not lower triangular.

DATA LAYOUT AND DATA LOCALITY

As shown with KRIPKE in [7], the choice of data layout, how the six dimensional angular flux phase space is mapped to a computer's linear memory space, greatly affects radiation transport simulation performance. In KRIPKE, unknowns are stored in a three level, nested fashion; a user chooses whether energy groups (G), directions (D), or spatial zones (Z) are at any particular level of the nesting. KRIPKE supports all six possible memory data layouts. Additionally, computational tasks like the sweep, scattering operator, etc. , in KRIPKE are completed by looping over all subdomains. A subdomain is one piece of the transport phase space that includes a contiguous subset of spatial zones, at least one energy group, and at least one direction.

The ISCB scheme was initially implemented in ARDRA [6] and as a result, inherited a GDZ data layout customized for DD and subdomains with a single energy group and direction. However, using a data grouping customized for DD, the 8 unknowns of a single 3-D super zone ISCB solve reside in 4 non-contiguous portions of memory. With larger subdomains and different data layouts, 8 different, non-contiguous memory locations would be accessed for each ISCB super zone solve. To demonstrate this, we apply ISCB to a 2-D problem, for a subdomain with two angles, two energy groups, and 8 spatial zones, requiring a total of 8 super zone solves to find all angular flux unknowns. Laying the data out in the ZDG pattern, ISCB using a diamond difference memory (DDM) grouping accesses data as shown in Fig. 1, where dashed boxes represent

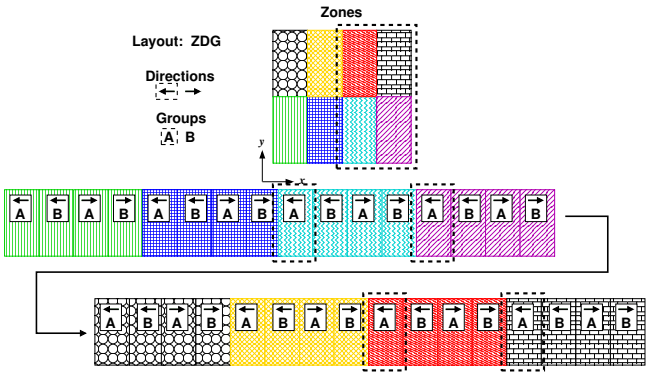

Fig. 1. DDM grouping for large subdomain ZDG data layout.

data access locations. Alternatively, we could group the data to be continuous for a given super zone rather than contiguous for each zone, yielding a contiguous memory (CM) access pattern as shown in Fig. 2.

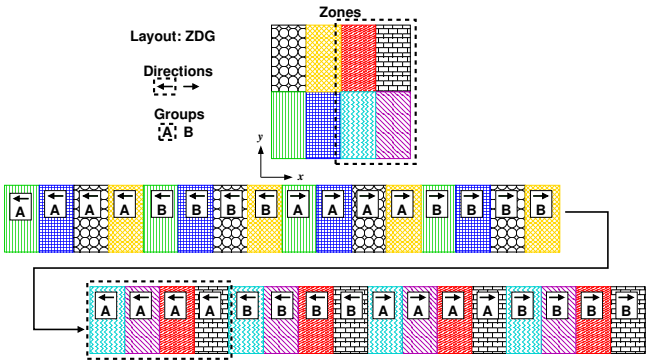

Fig. 2. CM grouping for large subdomain ZDG data layout.

ISCB MATRIX INVERSION

Numbering the corners of a given ISCB super zone solve, as shown in Fig. 3, yields a matrix, $\mathbf{L_1}$, with a non-zero pattern, as shown in Eq. (3), where X represents a non-zero number. There are two straightforward software options for solving

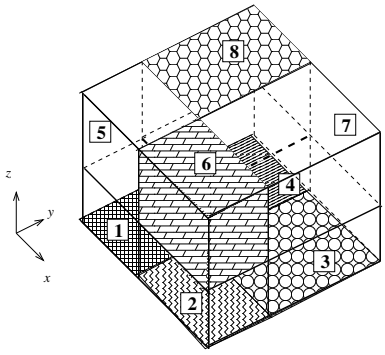

Fig. 3. Numbering of ISCB super zone corners.

the ISCB linear system of equations described by Eq. (3) via Gaussian elimination. The first and simplest to implement, is to use a third party library to solve the system of equations. We use the Intel MKL implementation of the LAPACK [8] `dgesv()` partial pivoting Gaussian elimination solve. The second option, a hand coded Gaussian elimination solve, requires a greater amount of developer work, but,

1. accounts for the sparsity pattern of the ISCB matrix,
2. does not perform any partial pivoting, and
3. does not perform any error/argument checking,

possibly leading to increased computational performance.

$$\mathbf{L_1} = \begin{bmatrix} X & X & \cdot & X & X & \cdot & \cdot & \cdot \\ X & X & X & \cdot & \cdot & X & \cdot & \cdot \\ \cdot & X & X & X & \cdot & \cdot & X & \cdot \\ X & \cdot & X & X & \cdot & \cdot & \cdot & X \\ X & \cdot & \cdot & \cdot & X & X & \cdot & X \\ \cdot & X & \cdot & \cdot & X & X & X & \cdot \\ \cdot & \cdot & X & \cdot & \cdot & X & X & X \\ \cdot & \cdot & \cdot & X & X & \cdot & X & X \end{bmatrix} \tag{3}$$

Additional options for solving the ISCB system of equations are possible if we re-order the ISCB matrix to minimize fill-in. Using the symmetric minimum degree re-ordering described in Table I, yields an ISCB matrix, $\mathbf{L_2}$, with non-zero structure as shown in Eq. (4).

TABLE I. Corner re-numbering to reduce matrix fill-in.

Re-ordering	1	2	3	4	5	6	7	8
Original ordering	8	1	3	6	2	4	5	7

$$\mathbf{L_2} = \begin{bmatrix} X & \cdot & \cdot & \cdot & \cdot & X & X & X \\ \cdot & X & \cdot & \cdot & X & X & X & \cdot \\ \cdot & \cdot & X & \cdot & X & X & \cdot & X \\ \cdot & \cdot & \cdot & X & X & \cdot & X & X \\ \cdot & X & X & X & X & \cdot & \cdot & \cdot \\ X & X & X & \cdot & \cdot & X & \cdot & \cdot \\ X & X & \cdot & X & \cdot & \cdot & X & \cdot \\ X & \cdot & X & X & \cdot & \cdot & \cdot & X \end{bmatrix}. \tag{4}$$

Again, we may apply Gaussian elimination to invert $\mathbf{L_2}$ for a given super zone solve. Alternatively, we could use a Schur complement solve. To use the Schur complement solve, we write the ISCB super zone solve described by $\mathbf{L}_2$ as a block system of equations:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} \vec{a} \\ \vec{b} \end{bmatrix}. \tag{5}$$

Equation (5) can be expanded as

$$\mathbf{A}\vec{x} + \mathbf{B}\vec{y} = \vec{a} \tag{6a}$$

$$\mathbf{C}\vec{x} + \mathbf{D}\vec{y} = \vec{b}. \tag{6b}$$

Solving Eq. (6b) for $\vec{y}$,

$$\vec{y} = \mathbf{D}^{-1}\left(\vec{b} - \mathbf{C}\vec{x}\right), \tag{7}$$

and inserting into Eq. (6a) yields:

$$\mathbf{A}\vec{x} + \mathbf{B}\mathbf{D}^{-1}\left(\vec{b} - \mathbf{C}\vec{x}\right) = \vec{a}. \tag{8}$$

Manipulating Eq. (8), we have:

$$\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)\vec{x} = \vec{a} - \mathbf{B}\mathbf{D}^{-1}\vec{b}. \tag{9}$$

The linear algebra required to find $\vec{x}$ via Eq. (9) can be accomplished using single instances of the LAPACK `dgemv()` [dense matrix-vector multiplication], `dgemm()` [dense matrix-matrix multiplication], and `dgesv()` [dense Gaussian elimination with partial pivoting] operations. Because $\mathbf{D}$ is diagonal, it is trivial to calculate and multiply by $\mathbf{D}^{-1}$. Likewise, since $\mathbf{A}$ is diagonal, it is also trivial to add $\mathbf{A}$ to the matrix $-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}$. Having obtained $\vec{x}$, we solve for $\vec{y}$ using Eq. (7) and a single LAPACK `dgemv()` operation.

COMPUTATIONAL RESULTS

To compare the performance of our various ISCB implementations, we conduct a weak, on node scaling study using Lawrence Livermore's RZMerl using KRIPKE [7]. A single node on RZMerl consists of two Intel Xeon E5-2670 processors with a total of 16 cores. Our scaling problems uses S_8 quadrature (96 total angles), 32 energy groups, and we scale the number of spatial zones per core, using 16 x 16 x 16 zones per core, requiring a total of $96 \times 32 \times 8 \times 8 \times 8$ ISCB super zone solves per core. We consider two subdomain sizes, small- with 1 group and 1 direction per subdomain and large- with 32 groups and 12 directions per subdomain.

TABLE II. Notation for different ISCB solution techniques.

Method	Matrix Ordering	Solution Method
DDM	Original	GE w/ `dgesv()`
CM	Original	GE w/`dgesv()`
HGE	Original	GE by hand
ROGE	Re-ordered	GE by hand
SC1	Re-ordered	SC by hand
SC2	Re-ordered	SC w/ `dgemv()`
SC3	Re-ordered	SC w/ `dgemv()`, `dgemm()`
SC4	Re-ordered	SC w/ `dgemv()`, `dgemm()` , `dgesv()`

In the results that follow, we consider several variants of the ISCB solution, and refer to each by the notation of Table II. In Table II, the DDM scheme uses the diamond difference memory grouping, all other methods use the contiguous memory ISCB memory grouping; methods either use Gaussian elimination (GE) or the Schur complement (SC) inversion. Additionally, "by hand" means that all linear algebra operations have been hard coded and loops have been unrolled by the developer.

Initial Comparisons

If data locality is the performance bottleneck of the ISCB scheme in ARDRA, we expect to see a significant performance increase (runtime decrease) when using the CM scheme vs. the DDM scheme, with the biggest runtime difference occurring for large subdomain sizes with the ZGD or ZDG data layout. In Fig. 4, we give the sweep times of the DDM and CM schemes for a ZDG data layout with large subdomains.

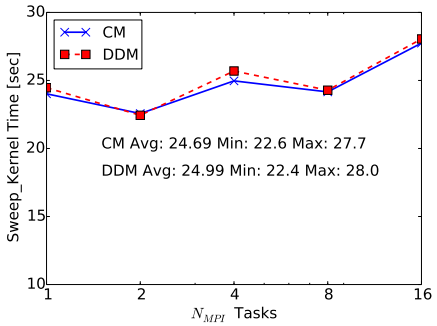

Fig. 4. CM and DDM timing with ZDG and large subdomains.

The Intel XeonE5-2670 has 32 KB of L1 cache and 256 KB of L2 cache per core, whereas the distance between farthest elements of a single super zone solve is 1092 KB for our chosen problem. Despite significantly more L1 and L2 cache misses than the CM scheme, the DDM scheme achieves effectively the same sweep runtimes, implying a performance bottleneck other than data locality, likely the 8×8 matrix inversion.

Variations of the Schur Complement Solve

In Fig. 5 we compare the performance of the Schur complement solve using various mixtures of LAPACK library calls and hand coded linear algebra operations. Though simplest from a coding perspective to use LAPACK function calls, the best performance is achieved when all linear algebra operations are written explicitly by hand.

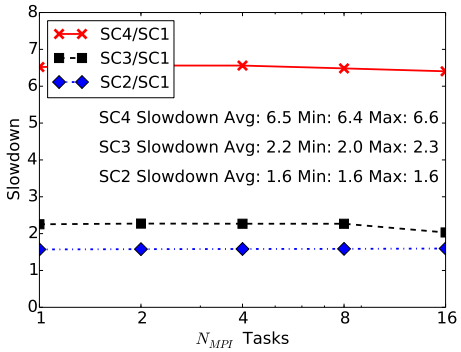

Fig. 5. Comparing LAPACK library functions vs. hand coding for the Schur complement solve. Results are for the GDZ data layout with small subdomains, but are representative of all data layouts and subdomain sizes.

Comparison of Optimized Solves

We now compare the performance of the fastest variations of the different ISCB solution techniques and data layouts. The results of Fig. 6 are for the GDZ data layout with small subdomains, but the results are reflective of all data layouts and subdomain sizes. For all data layouts and subdomain sizes,

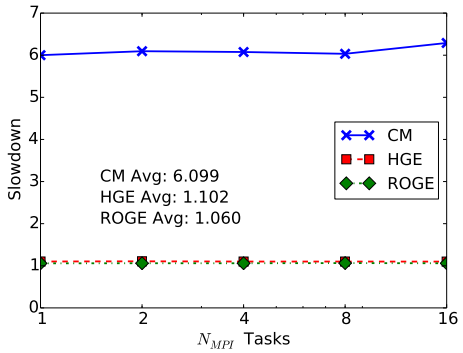

Fig. 6. Slowdown of different methods compared to SC1.

the SC1 method is the fastest, but the hand coded Gaussian elimination solves are only about 5-10% slower than SC1.

Comparison to DD

Finally, we compare the performance of SC1 to DD on the same mesh. As seen in Fig. 7, the time to solution of the SC1 scheme can be within a factor of two of DD for small subdomains or any data layout where zones are the innermost data nesting. However, Fig. 7 also illustrates one of the biggest performance advantages of DD on modern architectures, compiler can readily SIMD-ize the DD spatial discretization, allowing for full utilization of the CPU to process multiple data elements at once. The ISCB solve cannot be SIMD-ized due to the required matrix inversion. SIMD-ization would require storing multiple matrices (one for each group and direction) and and re-writing the matrix inversion to allow for multiple simultaneous matrix solves of similar, but distinct data.

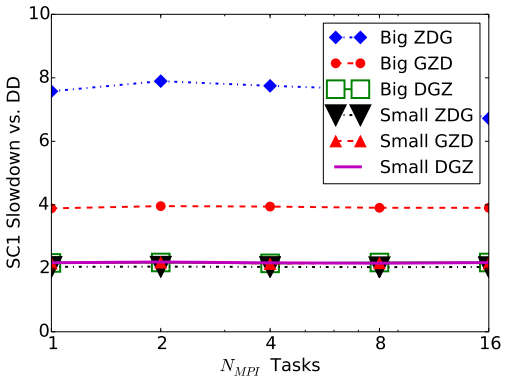

Fig. 7. SC1 slowdown vs to DD for equal unknown counts.

CONCLUSIONS

The ISCB spatial discretization is a more FLOP intensive spatial discretization than DD with the same memory footprint as DD. Though designed to maximize FLOP intensity, third party libraries such as LAPACK are not tuned for the unique, frequent, small linear algebra needs of neutron transport problems. While hand coding a sparse Gauss elimination is likely to be faster than a dense matrix solve (CM vs. HGE), were LAPACK suitable for small linear algebra problems, we would not have seen a performance increase with decreased numbers of LAPACK calls for the Schur complement solve (SC4 vs. SC1). Significant effort has reduced the relative compute time of ISCB to DD, but on current computer architectures DD is still faster than ISCB. Future work comparing the overall efficiency of ISCB to DD may reveal that while ISCB is slower than DD on current computers, ISCB is more accurate than DD by a factor greater than the compute time slowdown. As new computer architectures become available, we will test our hypothesis that neutron transport simulations will be bandwidth limited, ideally rendering the additional FLOPs required by ISCB to be "free" [in terms of time to solution].

ACKNOWLEDGMENTS

The authors are indebted to Britton Chang for his discussions regarding the reordering of the ISCB matrix and the Schur complement matrix solve. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

REFERENCES

1. E. E. LEWIS and W. F. MILLER, *Computational Methods of Neutron Transport*, ANS, La Grange Park, IL (1993).
2. E. W. LARSEN and J. E. MOREL, "Asymptotic Solutions of Numerical Transport Problems in Optically Thick, Diffusive Regimes II," *JCP*, **83**, 212–236 (1989).
3. S. H. LANGER, I. KARLIN, ET AL., "Performance Analysis and Optimization for BLAST, a Higher Order Finite Element Hydro Code," Tech. Rep. LLNL-PROC-666382, Lawrence Livermore National Lab (2015).
4. M. L. ADAMS, "Subcell Balance Methods for Radiative Transfer on Arbitrary Grids," *Transport Theory and Statistical Physics*, **26**, *4-5*, 385–431 (1997).
5. B. L. BIHARI and P. N. BROWN, "A Linear Algebraic Analysis of Diffusion Synthetic Acceleration for the Boltzmann Transport Equations II: The Simple Corner Balance Method," *SIAM J. Num. Analysis*, **47**, *3*, 1782–1826 (2009).
6. U. HANEBUTTE and P. N. BROWN, "ARDRA, Scalable Parallel Code System to Perform Neutron and Radiation Transport Calculations," UCRL-TB-132078, Lawrence Livermore National Lab (1999).
7. A. J. KUNEN, T. S. BAILEY, and P. N. BROWN, "KRIPKE- A Massively Parallel Transport Mini-App," in "Joint International Conference on Mathematics and Computations, Supercomputing in Nuclear Applications, and the Monte Carlo Method," ANS, LaGrange Park, IL (2015).
8. E. ANDERSON ET AL., *LAPACK Users' Guide*, SIAM, Philadelphia, PA, third ed. (1999).